Claims: We claim:

1) An unsolicited message rejecting communications processor connected to message transfer agents

MTA_0 with an Internet address of IP_0, from-address A_0, declared domain of D_0, and actual domain of DD_0, and

MTA_1 with an Internet address of IP_1 and to-address A_1

comprising:

a) monitoring means for monitoring the communications between MTA_0 and MTA_1;
b) determining means for determining if the communications contains an unsolicited message; and
c) intercepting means for intercepting a .\r\n end-of-message indicator reply from MTA_0, forcing MTA_0 to QUIT its connection with MTA_1 by sending an error reply to MTA_0 if the message is determined to be unsolicited.

whereby MTA_1 controls the interaction between MTA_0 and MTA_1 before a .\r\n end-of-message indicator reply from MTA_0 is received.

2) The unsolicited message blocking communications processor in Claim 1, further includes a allow_address database and wherein the determining means determines if a message is not unsolicited by checking if the IP_0 is in the allow_address database.
3) The unsolicited message blocking communications processor in Claim 1, further includes a prevent_address database and wherein the determining means determines if a message is unsolicited by checking if IP_0 is in the prevent_address database.
4) The unsolicited message blocking communications processor in Claim 1, further includes access to a open relay database and wherein the determining means determines if a message is unsolicited by checking if IP_0 is in the open relay database.
5) The unsolicited message blocking communications processor in Claim 1, further includes access to a DNS (domain name server) database and wherein the determining means determines if a message is unsolicited by checking if IP_0 has a domain name entry DD_0 in the DNS database.

6) The unsolicited message blocking communications processor in Claim 1, further includes a bad_from database and wherein the determining means determines if a message is unsolicited by checking if the from-address A_0 is in the bad_from database.

7) The unsolicited message blocking communications processor in Claim 1, further includes a suspect_domain database and wherein the determining means determines if a message is unsolicited by checking if the actual domain DD_0 matches the domain of from-address A_0 and the domain of from-address A_0 is in the suspect_domain database.

8) The unsolicited message blocking communications processor in Claim 1, wherein the determining means determines if a message is unsolicited by checking if the from-address A_0 matches the to-address (A_1).

9) The unsolicited message blocking communications processor in Claim 1, further includes a no_filter database and wherein the determining means determines if the message is to be blocked if it is determined to be unsolicited.

10) The unsolicited message blocking communications processor in Claim 1, wherein the determining means determines if a message is unsolicited by checking if the declared domain D_0 of MTA_0 is the same as the domain D_1 of MTA_1.

11) The unsolicited message blocking communications processor in Claim 1, wherein the determining means determines if a message is unsolicited by checking if the declared domain D_0 of MTA_0 does not match the real domain DD_1 and the declared domain D_0 is in the suspect_domain database.

12) The unsolicited message blocking communications processor in Claim 1, further includes a bad_word database and wherein the determining means determines if a message is unsolicited by checking if the subject line of the message contains any words in the bad_word database.

13) The unsolicited message blocking communications processor in Claim 1, further includes a bad_fingerprint database and wherein the determining means determines if the hash "fingerprint" of a portion of the message is in the bad_fingerprint database.

14) The unsolicited message blocking communications processor in Claim 1, further includes a rejected_connection database which logs the time, from-address A_0, to-

address A_1, and the reason for the rejection if a message is rejected if the message is determined to be unsolicited.

15) The unsolicited message blocking communications processor in Claim 1, further includes a allowed_connection database which logs the time and to-address A_1 if the message is determine not to be unsolicited.

16) A method for

a receiving networked computer system with an Internet connection, a mail transport agent MTA_1, an Internet address IP_1, to-address A_1, and an operating system capable of executing the method

to reject unsolicited messages from

a transmitting networked computer system with an Internet connection and a message transfer agent MTA_0, an Internet address IP_0, from-address A_0, declared domain D_0, and actual domain DD_0

comprising the steps of:

a) waiting for a new SMTP connection request;
b) relaying and monitoring the replies from MTA_0 to MTA_1;
c) relaying replies from MTA_1 to MTA_0;
d) intercepting the .\r\n end-of-message indicator reply from MTA_0 to MTA_1;
e) determining if the message is unsolicited by analyzing the monitored replies;
f) releasing the intercepted .\r\n end-of-message reply if the message is determined not to be unsolicited; and
g) sending a error reply to MTA_0 to force MTA_0 and MTA_1 to close down their connection;

whereby MTA_1 controls the interaction between MTA_0 and MTA_1 until a .\r\n end-of-message indicator reply is received from MTA_0.

17) A method for

a receiving networked computer system with an Internet connection, DNS server, and open relay database, a mail transport agent MTA_1, IP address IP_1, a domain name D_1, a recipient, A_1, an allow_address database, a prevent_address database, a suspect_domain database, a bad_from database, a no_filter database, a yes_filter database, a bad_word database, a bad_fingerprint, a rejected_connection database, an allowed_connection database, and an operating system capable of executing the method

to reject unsolicited messages from

a transmitting networked computer system with an Internet connection, a message transfer agent MTA_0, an IP address of IP_0, a declared domain name D_0, a real domain name DD_0, and a sender address of A_0

comprising the steps of:

a) waiting for a SMTP connection request on the receiving networked computer system's Internet connection;

b) sending a 220 reply to MTA_0 to acknowledge the requested connection;

c) extracting IP address IP_0 from the connection request;

d) requesting a domain name DD_0 for IP_0 from the DNS server;

e) testing if domain name DD_0 is "no name";

f) testing if IP_0 is in an open relay database;

g) testing if IP_0 is in the allow_address database;

h) testing if IP_0 is in the prevent_address database,

i) requesting a connection with MTA_1;

j) waiting for a 220 reply from MTA_1 to acknowledge the requested connection;

k) waiting for a reply from either MTA_0 or MTA_1;

l) jumping to step o) if the reply is not from MTA_1;

m) relaying the reply from MTA_1 to MTA_0;

n) jumping to step k) to wait for a new reply;

o) jumping to step u) if the reply from MTA_0 is not a **HELO**;

p) extracting domain D_0 from the reply;

q) testing if declared domain D_0 of MTA_0 matches domain D_1 of MTA_1;

r) testing if declared domain D_0 does not match real domain DD_0 of MTA_0 AND declared domain D_0 is in the suspect_domain database;

s) relaying the HELO reply from MTA_0 to MTA_1;

t) jumping to step k) to wait for a new reply;

u) jumping to step aa) if reply from MTA_0 is not a **MAIL**;

v) extracting from-address A_0;

w) testing if A_0 is in the bad_from database;

x) testing if DD_0 does not match the domain of A_0 and the domain of A_0 is in the suspect_domain database;

y) relaying MAIL reply to MTA_1;

z) jumping to step k) to wait for a new reply;

aa) jumping to step ii) if reply from MTA_0 is not a **RCPT**;

bb) extracting to-address A_1;

cc) testing if A_1 is in no_filter database;

dd) testing if A_0 matches A_1;

ee) testing if A_0 is in the no_filter database;

ff) testing if A_0 is in the yes_filter database;

gg) relaying RCPT reply to MTA_1;

hh) jumping to step k) to wait for a new reply;

ii) jumping to step yy) if reply from MTA_0 is not **DATA**;

jj) relaying DATA to MTA_1;

kk) waiting for 354 reply from MTA_1;

ll) relaying 343 reply to MTA_0;

mm) wait for body of message;

nn) relaying body of message to MTA_1;

oo) waiting for .\r\n end-of-message indicator;

pp) testing if any word in the subject line of the message is in the bad_word database;

qq) testing if the hash "fingerprint" of a portion of the message is in the bad_fingerprint database;

rr) jumping to step vv) if NOT(t_allow OR t_no_filter OR OR NOT t_yes_filter OR NOT (t_prevent OR t_open OR t_DD-) OR t_bad_from OR t_suspect_domain OR t_echo_domain OR t_forged_domain OR t_bad_word OR t_bad_fingerpring)) ;

ss) logging time and to-address A_1 in the allowed_connection database;

tt) relaying the .\r\n end-of-message indicator reply to MTA_1 to continue the conversation;

uu) jumping to step k) to wait for a new reply;

vv) logging the time, from-address A_0, to-address A_1, and the reason for rejecting the connection in the rejected_connection database;

ww) sending a 554 reply to MTA_0 to terminate the conversation;

xx) jumping to step k) to wait for a new reply;

yy) jumping to step ggg) if reply from MTA_0 is not **RSET, SEND, SCML, SAML, VRFY, NOOP, EXPN, HELP,** or **TURN;**

zz) relaying reply to MTA_1;

aaa) jumping to step j) to wait for a new reply;

bbb) jumping to step ddd) if reply from MTA_0 is not a **QUIT**;

ccc) relaying the QUIT reply to MTA_1;

ddd) waiting for 221 reply from MTA_1

eee) relaying 221 reply from MTA_1 to MTA_0;

fff) jumping to step a) to wait for a new connection;

ggg) sending a 500 reply to MTA_0 to signal a syntax error; and

hhh) jumping to step a) to wait for a new connection.